# WEST Search History

```
DATE:  Thursday, November 20, 2003
```

| **Set Name** side by side | **Query** | **Hit Count** | **Set Name** result set |
|---|---|---|---|
| *DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ* | | | |
| L8 | L7 and (automatic or automatically or automated) | 55 | L8 |
| L7 | L6 and @AD<19990922 | 80 | L7 |
| L6 | (update or updating or upgrade or upgrading) near8 software near8 network near8 server | 182 | L6 |
| L5 | L4 and @AD<19990922 | 62 | L5 |
| L4 | (update or updating) near8 software near8 network near8 server | 143 | L4 |
| L3 | L2 and ((update or updating) near8 software) | 0 | L3 |
| L2 | L1 and @AD<19990922 | 11 | L2 |
| L1 | automated adj4 collection adj4 device | 31 | L1 |

```
END OF SEARCH HISTORY
```

WEST

Generate Collection

L8: Entry 2 of 55 File: USPT Sep 2, 2003

DOCUMENT-IDENTIFIER: US 6614804 B1
TITLE: Method and apparatus for remote update of clients by a server via broadcast satellite

Application Filing Date (1):
19990322

Brief Summary Text (9):
With the current WEBTV system, supplemental services data is downloaded in advance during low-use periods to the extent possible. For instance, WEBTV client terminals are configured to automatically call in to the server during the night to check for email and receive other information. However, because this process must be individually repeated for each client terminal, substantial telephone costs may be incurred by the subscription user or the service provider or both. In addition, in some cases, the quantity of data to be retrieved is so large that long periods of time are required to complete the download.

Detailed Description Text (12):
It also will be appreciated that part of a software upgrade download may include executable instructions or code that represents a script that, when executed by a client, automatically reconfigures the client. An example of such a configuration script would be a `reboot` or `power-up` script that, once executed, changes the operational parameters of the home entertainment device.

Detailed Description Text (29):
FIG. 5 charts the method of the invention by which video data downloads or software upgrades are conveyed to the clients or subscribers. The invented conveyance method by which a central server downloads video content or software upgrades to a client network having plural distributed clients may be seen to include the following steps. The method involves, at 200, first downloading, from the satellite broadcast link, information transmitted onto the satellite broadcast network from the server processing system, and, at 202, writing the downloaded information into a programmable nonvolatile memory in the client network to update information previously stored in the programmable non-volatile memory.

WEST

Generate Collection

L8: Entry 3 of 55 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591290 B1
TITLE: Distributed network application management system

Application Filing Date (1):
19990824

Detailed Description Text (32):
In this environment, 1. Automatic software update--software updates are transparently available from the distributed network application management server for the browser and Email functions. 2. Enhanced Performance--distributed network application management architecture relies on the Inferno name space and network operating system to make network resources transparently available to user applications. 3. Dynamic Updates--access to new or enhanced features including new attachment formats for email, new plug-ins for the Web browser are available to the user applications as they arrive on the distributed network application management server. 4. Personalization--the distributed network application management server provides end user personalized home page, preferences and information delivery. 5. Mobility--the distributed network application management server provides each end user with a uniform (personalized) environment independent of the physical location of the user log-in. 6. Personal Storage--the distributed network application management server provides limited user data storage consistently accessible from anywhere to store Email messages, Web browser bookmarks, telephony address book in the network file system.

Detailed Description Text (40):
Persistent user data is mobile since such user data is provided locally from the user home server (appears in the local application name space). Such user data is defined, used or changed by the application as if it were local to the device. A persistent communication path is also automatically re-established by the client in case of failure, as if the user migrated from device 131 to itself.

WEST

☐ Generate Collection

L8: Entry 4 of 55 File: USPT Jul 1, 2003

DOCUMENT-IDENTIFIER: US 6587684 B1
TITLE: Digital wireless telephone system for downloading software to a digital telephone using wireless data link protocol

Abstract Text (1):
A digital wireless telephone downloads software related to digital telephone services using a client browser. The digital telephone initiates a data call to an interworking unit via a digital wireless telephone network, using a prescribed wireless data protocol such as IS-95A. The interworking unit recovers the payload of the wireless data packets to establish a two-way data link with the digital telephone. The interworking unit sends data messages to a destination server across a second two-way data link in a packet switched network to establish a two way session between the digital telephone and the destination server. The user of the digital telephone can thus communicate with the server via a two-way application-layer session using hypertext-based messaging. The digital telephone can thus navigate between different servers on the packet switched network for activation of different digital telephone services, and for downloading new software or updating existing software related to the digital telephone services.

Application Filing Date (1):
19980728

Detailed Description Text (8):
As shown in FIG. 1, the proxy gateway server 20 is connected to a TCP/IP based packet switched network 22, such as a private intranet or the Internet. The packet switched network 22 enables the proxy gateway 20 to selectively connect the digital telephone 16 to at least one of a plurality of different servers. Each server in the packet switched network is configured for downloading a particular software for activation or upgrading of a particular digital telephone service. For example, the packet switched network 22 includes a provisioning server 24,a revision control server 26,and a user database 28 configured for storing mobile user information, described below.

Detailed Description Text (59):
The digital telephone 16 then accesses the activation home page of the provisioning server 24 in step 136 using the supplied URL via the secure link. The activation server 24 establishes a two-way application layer session 62 with the digital telephone 16 by sending a welcome message in step 138 that prompts for the user authentication code. The application layer 60a may then automatically supply the mobile identification number (MIN) and/or user authentication code in encrypted format in step 140 to the provisioning server 24, or the user may manually input the authentication code.

WEST

Generate Collection

L8: Entry 5 of 55 File: USPT Apr 29, 2003

DOCUMENT-IDENTIFIER: US 6557169 B1
TITLE: Method and system for changing the operating system of a workstation connected to a data transmission network

Application Filing Date (1):
19990323

Brief Summary Text (2):
The present invention relates to using software distribution server(s) on a data transmission network for updating application software of a workstation connected to the network, and relates in particular to a method and system for changing the operating system of a workstation connected to a data transmission network.

Detailed Description Text (24):
In conclusion, the method according to the present invention which is described above in reference to FIGS. 1 and 2 presents many advantages. In particular, it reduces the installation time since the change of the operating system is made automatically: it saves manpower cost since there is no need to dispatch on-site people; and it enables scheduling installation at the most convenient time (off-peak periods during nights or weekend) when the operation of the workstation can be interrupted or when human intervention (the alternatives) would be expensive.

WEST

Generate Collection

L8: Entry 6 of 55 File: USPT Aug 13, 2002

DOCUMENT-IDENTIFIER: US 6434745 B1
TITLE: Customized web browsing and marketing software with local events statistics database

Abstract Text (1):
The invention includes customized software to be used on an end-user computer having fixed storage and an interface with a network server using a network. A GUI component controls a browser which accesses publisher content that is at least partially installed on the fixed storage. Software is provided for monitoring end-user behavior, on-line and off-line, and maintaining a local event statistics database of the end-user behavior including event data related to end-user events. Software is provided for detecting a network connection and transmitting at least a portion of the local event statistics database to the network server. Software is provided for compiling summary information from the local event statistics database wherein the summary information comprises the portion of the local event statistics database. E-mail grabbing software automatically examines each URL loaded in the browser and locates and saves E-mail addresses, URL, title pages, last modified date, and any keywords present in a <META> tag of the E-mail URL to a searchable local E-mail database on the end-user's computer. An analysis program on the network server receives and analyzes local event data from the local event statistics databases and stores results from the analysis program into a statistics database on a statistics server accessible through the network server.

Application Filing Date (1):
19990915

Brief Summary Text (13):
A password manager means allows the end-user to record a script of a sequence of keystrokes performed to log-in to a password secured URL, to save the script to a local script database, generate a log-in script which the browser will run whenever it detects the end-user navigating to the password secured URL with a defined script, and automatically log the end-user in to the password secured URL.

Brief Summary Text (14):
The software includes an E-mail grabbing means for automatically examining each URL loaded in the browser to locate E-mail addresses and save the Email address, the URL, the title of the page of the URL, the URL's last modified date, and any keywords present in a <META> tag of the URL to a searchable local E-mail database on the end-user's computer. An E-mail interface means allows the end-user to search the E-mail database with one or more search terms and display E-mail information for E-mail addresses stored in the E-mail database which satisfy a query. The E-mail interface also allows the end-user to click-on a displayed E-mail address and invoke a default E-mail program stored on the end-user's computer to send an E-mail to the displayed E-mail address. The E-mail interface includes a button to activate a reporting means for saving the E-mail information to an E-mail file.

Drawing Description Text (13):
FIG. 10 is a flow chart illustrating an E-mail grabbing component of the browser for automatically examining each URL loaded in the browser to locate any E-mail addresses;

Detailed Description Text (3):
An installation program stored on the CD-ROM 12 is used by end-user to install the browser 8 and automatically stores publisher's content on the hard drive 20 on the end-user computer 14. The publisher's content stored on the hard drive 20 is accessible by the end-user with using the browser 8 which can use and display the content stored on either or both the hard drive and the CD-ROM. The installation program also installs a customized icon on a graphical end-user interface (GUI) which appears on the screen of the end-user computer 14 as illustrated in FIG. 3.

Detailed Description Text (5):
The browser includes an event statistics component 38 which is software for monitoring end-user behavior and maintaining a local event statistics database 40 of end-user behavior on the end-user computer 14 and, preferably, on the hard drive 20. The end-user behavior is determined by monitoring events that occur while the end-user uses the browser. The event data is stored in the end-user computers 14 and is automatically transmitted by the browser to the web server 24 when the end-user computer 14 is connected to the network 26. The end-user behavior is monitored any time the end-user uses the browser both on-line when connected to a network such as the World Wide Web (Internet) or intranet and off-line when not connected to a network.

Detailed Description Text (15):
FIG. 10 is a flow chart illustrating an E-mail grabbing component of the browser 8 for automatically examining each URL loaded in the browser to locate any E-mail addresses. When an E-mail address is located, that E-mail address, the URL, the title of the page, the URL's last modified date, and any keywords present in a <META> tag are automatically saved to a local E-mail database in the form of an E-mail DB (database) record as illustrated in an exemplary E-mail DB record in FIG. 11.

CLAIMS:

13. Software as claimed in claim 12 wherein said means for updating said publisher content includes downloading new files from the network server and storing them on the end-user's computer and deleting existing files which are stored on the end-user's computer.

16. Software as claimed in claim 14 further comprising a password manager means for allowing the end-user to save a script of a sequence of log-on keystrokes performed to log-in to a password secured URL, to save said script to a local script database, generate a log-in script from said saved script of a sequence of log-on keystrokes which the browser will run whenever it detects the end-user navigating to said password secured URL with a defined script, and automatically log the end-user in to said password secured URL.

18. Software as claimed in claim 17 further comprising an E-mail grabbing means for automatically examining each URL loaded in the browser to locate E-mail addresses.

19. Software as claimed in claim 5 further comprising an E-mail grabbing means for automatically examining each one of URLs loaded in the browser to locate E-mail addresses and saving the E-mail address, the URL, the title of the page of the URL, the URL's last modified date, and any keywords present in a <META> tag of the URL to a searchable local E-mail database on the end-user's computer.

32. Software as claimed in claim 30 further comprising updating means for detecting presence of a connection to the network during start-up of the browser software, executing a transaction to the server to determine if the icon should be modified, and means for updating or changing the customized icon when the end-user's computer is connected to the server over the network which issues a set of updating instructions to the end-user's computer to update or change the icon.

44. Password managing software for use with a browser software and stored in machine readable format, said password managing software comprising: a recording means for an end-user to record a script of a sequence of keystrokes performed to log-in to a password secured address on a computer system; a means for saving the script to a local script database, a means for generating a log-in script from said script in said local script database which the browser software runs whenever the browser software detects the end-user navigating to the password secured address; and a means for automatically logging the end-user in to the password secured address.

WEST

Generate Collection

L8: Entry 14 of 55 File: USPT Dec 26, 2000

DOCUMENT-IDENTIFIER: US 6167567 A
TITLE: Technique for automatically updating software stored on a client computer in a networked client-server environment

Abstract Text (1):
A technique for automatically updating software, including but not limited to application programs, residing on, e.g., a client computer. Specifically, an update script is stored on a network server for each software product to be updated and, where appropriate, for each different country or locale in which that product will be installed. At a scheduled time, the client computer automatically, through an executing updating application: establishes a network connection to the server; constructs a file name for a file containing an appropriate update script; and then downloads that file from the server. The script contains appropriate update information, including whether the update is to occur through a web site or through the script, and if the latter, listings of operating system (O/S) specific and O/S-independent product update files. For a script-based update, the updating application downloads those update files, as specified by the script, corresponding to the executing O/S and then, in a sequence specified in the script, executes various files therein to complete the update. Once the update successfully concludes, the updating application appropriately updates the locally stored version number of the installed software and schedules the next update accordingly.

Application Filing Date (1):
19980505

Brief Summary Text (3):
The invention relates to a technique, specifically apparatus and accompanying methods, for automatically updating, in a networked client-server environment, software stored on, e.g., a client computer. Inasmuch as this software includes, but is not limited to application software, use of the invention can advantageously and significantly reduce costs and simplify tasks associated with maintaining and administering client software across essentially all types of different client-server environments.

Brief Summary Text (8):
One conventional widely-used approach aimed at reducing the cost of maintaining software has been to post software updates on a network server and permit users to access, download and remotely install a desired update(s) from the server onto his(her) client computer. This approach is frequently used by device manufacturers, such as those manufacturing modems, video driver boards and other computer peripherals, who post their updates to their FTP or web servers which, in turn, can be remotely accessed by their customers through the Internet. The same approach is often used by software manufacturers to post "patches" and maintenance updates (so-called "service packs") for access by their user community.

Brief Summary Text (11):
The art, sensing the difficulties users encounter in updating their client software, is starting to provide products with an automatic software update capability. In that regard, once a product is installed and is executing at a client computer, this capability generally involves establishing, either on user request or automatically on a time-scheduled basis, a network connection from the client computer to an FTP site for the manufacturer of that product; then, determining, typically based on version numbers of the most recent update available at that site and installed at the client, whether the client software should be updated; if the client is to be updated, downloading the update file(s) from the ftp site; and, finally, executing an appropriate installation program to install the update and change the version number of the client software.

Brief Summary Text (14):
Thus, a need exists in the art for a technique, specifically apparatus and accompanying methods, for automatically updating software installed on a client computer. This technique should be able to update software from a wide variety of sources, e.g., manufacturers, and types, i.e., not just application programs but also, e.g., device firmware. In addition, this technique should be able to select, where appropriate, a correct update for the country and/or locale at which the client computer is situated. Advantageously, use of such a technique should substantially simplify the task of maintaining client computers, particular in terms of correctly updating their client software, and appreciably reduce its attendant cost; as well as reducing a burden and associated cost which a manufacturer faces in propagating software updates throughout its user community.

Brief Summary Text (16):
The present invention overcomes the deficiencies in the art and satisfies these needs through a client-based application that automatically and properly updates substantially any client-resident software via a networked server and without substantially any user intervention, but also, where necessary, to account for regional or local update differences.

Brief Summary Text (17):
Essentially and broadly speaking, to permit a community of client computers to automatically update installed software products, our inventive technique relies on storing an associated update script, on a network server, for each such product and, where appropriate, for each different country in which that product will be used. A client computer seeking to update any such product, such as at a scheduled time or through manual user initiation, will, through our inventive client updating application, construct the name of the file for the appropriate update script and then download that file, via a network connection, from the server and process it to perform the update.

Brief Summary Text (24):
As a feature of our invention, not only can our inventive technique update application software residing on a client, it can also update device firmware, device drivers, O/S modules and nearly any other software that resides on the client computer itself or on substantially any computer in the network, or device accessible to the network. In this case, all that is required is that the client computer have suitable network access to the specific software that is to be updated and that the software appropriately register itself, for updating, with the client. Moreover, our technique is not limited to updating software that emanates from just one source, such as a specific product manufacturer, or just one product. In that regard, our technique presents common and rather simple programming interface requirements to which any third-party can adhere, e.g., creation of a specific product sub-key in an O/S registry and various entries therein. Multiple products, regardless of their individual sources, that satisfy these requirements, during their installation on a client computer, can be automatically updated by our technique.

Drawing Description Text (3):
FIG. 1 depicts a high-level simplified block diagram of client-server environment 5 in which illustratively client firmware is to be automatically updated by client PC 10, from server 70, in accordance with our inventive teachings;

Detailed Description Text (4):
FIG. 1 depicts a high-level simplified block diagram of client-server environment 5 in which illustratively client firmware is to be automatically updated by client personal computer (PC) 10, from server 70, in accordance with our inventive teachings. As shown, environment 5 is formed of client PC 10 which through terminal adapter 40 and leads 35 and 47, establishes a network connection, via network 50 (which can be the Internet, intranet or any other network) and leads 55, to server 70. Inasmuch as the particular implementation and architecture of network 50 and the specific communication modality supported by the terminal adapter (such as whether it is an ISDN device, analog modem or other mode of communication) for transport over the network are both irrelevant to the present invention, the ensuing discussion will omit all such details. Suffice it to say, terminal adapter 40 bi-directionally interfaces client PC 10 to network 50. This adapter also contains firmware 45 for which its manufacturer issues software updates from time to time. These updates may include, e.g., patches and other revisions, or constitute a new software release that provides enhanced functionality. Though device 40 is illustratively shown and described as an external terminal adapter, this device can alternatively be a router, hub, or generally any externally-connected product, or

any circuit board connected internally to the client PC, or located anywhere across a network to which client PC 10 has access, that contains or has associated with it software for which an update will become available.

Detailed Description Text (6):
Thusfar described, to permit conventional network-based updating of firmware 45 for terminal adapter 40, the manufacturer will place requisite update files either on its FTP site and/or its web site and permit its customers to access, through their client PCs, either site and download the necessary files. To do so, a user stationed at client PC 10 will typically establish either an FTP connection through network 50 to server 70, specifically to FTP server 82, or an HTTP connection to HTTP server 85. Having done so, the user will then navigate through the FTP site or web site to locate and download the proper update files. Thereafter, these files will then install the update either off-line, through user initiated-execution of typically an appropriate "setup" (setup.exe) file, or through automatic initiation through the web site. Once the update has completed, the client PC will then terminate the FTP or HTTP connection to server 70.

Detailed Description Text (8):
In contrast, our present invention not only permits substantially any client-resident software to be automatically and properly updated, through a networked server and without substantially any user intervention, but also, where necessary, to account for regional or locale update differences. Our present invention can readily function with any IP (Internet Protocol) network, regardless of whether a network connection is made through a dedicated local area network or dial-up connection.

Detailed Description Text (9):
Essentially, to permit a community of client computers to automatically update installed software products, our inventive technique relies on storing an associated update script, on a network server, for each such product and, where appropriate, for each different country in which that product will be installed. A client computer, such as client computer 10, seeking to update any such product, such as at a scheduled time, will automatically, through our inventive client updating application: (a) establish a network connection to the server, (b) construct the name of the file for the appropriate update script and then (c) download that file from the server. The script contains appropriate update information, including whether the update is to occur through a web site or through the script, and if the latter, listings, for the product, of operating system (O/S) specific update files (for various different operating systems which the product supports) and, where appropriate, of O/S-independent update files. If the update is to occur through the script, then the client updating application downloads, from the server, those update files, as specified by the script, corresponding to its own executing O/S and, in a sequence specified in the script, executes various files therein to complete the update. If the update for the product is to occur through a web site, a web browser, residing at the client computer, is launched by the client updating application and a URL of that site is passed to the browser. The user then interacts, through the browser, with the web site to update the product. Once the script- or web-based update successfully concludes, the client updating application appropriately updates the locally stored version number of the installed software and schedules the next update accordingly.

Detailed Description Text (15):
Now, with specific reference to the software updating example shown in FIG. 1, while software for terminal adapter 40 is installed into client PC 10, that software will establish suitable entries in the registry for updating terminal adapter firmware 45. At an appropriate time (or in response to user-initiation, if desired), updating application 500 will update this firmware, by downloading the specific update script and files as discussed above. Once all the update files have been suitably downloaded to client PC 10, the client PC, under control of application 500, will execute downloaded "run" files so as to generate and flash suitable updated firmware, as firmware 45, into terminal adapter 40. Thereafter, application 500 will change the version number of the firmware, for terminal adapter 40, stored in the registry and schedule the next automatic update for this particular firmware.

Detailed Description Text (28):
Configuration application 25 provides a graphical user interface (GUI) through which a user stationed at client PC 10 can interact with and configure application 500. The extent of this configuration, which will become evident from the screen displays shown in FIGS. 14A-14D, includes, e.g., detecting and registering any software product for use with application 500; enabling event logging of update activity; and for each

software product so registered, e.g.: scheduling update intervals for that product, specifying a network connection method and connection parameters for connecting to a network server for updating that product, and confirming all updates of that product with a user. Configuration application 25 stores and accesses, as symbolized by line 316, configuration information within local storage 310 which is collectively implemented by requisite storage space within hard disk 232 (see FIG. 2). Local store 310 includes, as shown in FIG. 3, O/S registry 247. Additionally, configuration application 25 also communicates, as symbolized by line 325 and through COM 320, with updating application 500.

Detailed Description Text (33):
First, through operations 402, either a user manually generates an update request, such as depressing an "Update Now" button on a specific configuration screen display for a desired product, specifically screen display 1430 shown in FIG. 14B, to update product i or a scheduled time occurs at which product i will be automatically updated. Next, through operations 404, application 500 executing at the client PC determines, through a conventional API (application program interface) call to client O/S 245 (see FIG. 2) to discern the country (or geographic locale) in which the client PC resides. This country was specified by the user of that PC typically during initial installation of its client O/S. Once application 500 discerns the country or locale, operations 406, as shown in FIGS. 4A and 4B, occur through which the client PC reads O/S registry 247 for a URL for an FTP site that contains the update script(s) for product i. Once this occurs, operations 412 occur to establish an FTP connection, as symbolized by line 410, to this site, specifically here to FTP server 82, and to a directory of update scripts residing at that server.

Detailed Description Text (41):
Upon occurrence of a user-configured date for a product update or a user-initiated update for that product, e.g., software product i, application 500 executes. Within this application, execution first proceeds to block 503 which reads a URL for an FTP update site for this product and then attempts to establish an FTP connection to this site. Thereafter, execution is directed to decision block 506 which determines whether the FTP connection was made. If the connection was not made, then this decision block routes execution via NO path 507 and path 532, to block 588. Block 588, when executed, automatically schedules a date for the next update for this product. Thereafter, execution proceeds to decision block 590 which, based on a configuration setting, determines whether the user is to confirm updates. If the user has required such confirmation, then decision block 590 routes execution, via YES path 591, to block 594. This latter block, when executed, notifies the user accordingly of the next scheduled update and requests the user to confirm it. Once the user so confirms the update, typically by clicking a displayed "OK" button with his mouse, execution proceeds to block 596 which logs an event, here the inability to connect to the FTP update site. Once the event is logged, execution exits from application 500. Alternatively, if the user has not requested update confirmation, then execution proceeds, via NO path 592 emanating from decision block 590, directly to block 596.

Detailed Description Text (45):
However, if an update is available, then decision block 519 routes execution, via YES path 520, to decision block 522. This latter decision block, based on the update confirmation configuration setting, determines whether the user is to confirm updates. If the user has required such confirmation, then decision block 522 routes execution, via YES path 524, to block 525. This latter block, when executed, displays the version number of the update that is available for installed product i and asks the user whether the update should now proceed or not. Thereafter, execution proceeds to decision block 528 to determine, based on a response from the user, whether the update is to occur now. If the user has indicated that the product should not be updated, typically by having clicked a then displayed "NO" or "CANCEL UPDATE" button (not specifically shown in the figures) with his(her) mouse, then decision block 528 routes execution, via NO path 530 and path 532, to block 588. This block, when executed, automatically schedules a date for the next update for this product, and so forth as described above, after which execution ultimately exits from application 500. Alternatively, if the user has indicated that the update should now proceed, typically by having clicked a then displayed "YES" button with his(her) mouse, decision block 528, routes execution, via YES path 529, to decision block 534. Execution also reaches this latter decision block, via NO path 523, emanating from decision block 522, in the event the update confirmation configuration setting indicates that the user does not want to confirm each update.

Detailed Description Text (47):

If the update is to proceed through the update script, then decision block 534 routes execution, via NO path 536, to block 537. This latter block, when executed, detects the specific O/S then operating in the client PC and processes those sections of the script which are: (a) O/S-independent, and (b) correspond to the detected O/S. This processing entails, given the parameters (as discussed below) in these sections, establishing appropriate lists of "copy" and "run" files to download and determining source directories on the FTP server at which these files are located and destination directories on the client PC into which these files are to be copied and, for the "run" files, executed. The file names for these "copy" and "run" files collectively define an "update file name" set with the corresponding update files themselves defining an "update file" set. Thereafter, execution proceeds to block 538. This block, when executed, downloads each of these "copy" files from its source directory on the FTP site into its destination directory on the client PC. Once this occurs, execution proceeds to decision block 539 which tests whether all the "copy" files were successfully downloaded. If the download of all the "copy" files did not succeed, e.g., a "copy" file could not be downloaded or an error arose during its downloading, then decision block 539 routes execution, via NO path 541, to block 542. If the FTP connection still exists to the update FTP site, block 542 simply terminates this connection. Thereafter, execution is directed, via path 532, to block 588 and so forth, as described above, after which execution ultimately exits from application 500. Alternatively, if all the "copy" files were successfully downloaded, then decision block 539 routes execution, via YES path 540, to block 543. This latter block, when executed, downloads each of the listed "run" files from its source directory on the FTP site into its destination directory on the client PC and then executes each of these files in the order downloaded. Once this occurs, execution proceeds to decision block 545 which tests whether all the "run" files were successfully processed, i.e., downloaded and executed. If the processing of these files did not succeed, e.g., a "run" file could not be downloaded or executed or an error arose during its downloading or execution, then decision block 545 routes execution, via NO path 546, to block 547. If the FTP connection still exists to the update FTP site, this latter block simply terminates this connection. Thereafter, execution is directed, via path 532, to block 588 and so forth, as described above, after which execution ultimately exits from application 500. Alternatively, if all the "run" files successfully processed, then decision block 545 routes execution, via YES path 548, to block 549. To the extent any "run" file is still executing, block 549 simply waits for the last such file to complete its execution. Once all such files have fully executed to install the update, then block 550 executes. This block, when executed, automatically schedules a date for the next update for product i. Thereafter, execution proceeds to decision block 552 which determines, based on the update confirmation configuration setting, whether the user required that (s)he be notified of each update and enter his(her) subsequent acknowledgement of its completion. If so, decision block 552 routes execution, via YES path 554, to block 556. This latter block, when executed, prompts the user to confirm, typically through a mouse click on a button then appearing on the display (display 283 in FIG. 2), that the update has completed. If the user then confirms that the update was completed, then decision block 558, shown in FIGS. 5A and 5B, routes execution, via YES path 560, to block 562. This latter block, when executed, updates the version number, stored in the registry, for product i with the version number provided in the update script. Once this occurs, block 563 determines whether the update script specifies that the user is to re-boot the client PC in order to complete the update. If such a re-boot is required, then this decision block routes execution, via YES path 565, to block 566. This latter block, when executed, displays a suitable notification to the user to re-boot the client PC. Once this occurs, execution proceeds to block 567. Alternatively, is such a re-boot is not required, then execution proceeds directly to block 567 via NO path 564 emanating from decision block 563. Block 567 then executes to terminate the FTP connection. Thereafter, execution proceeds, via path 568, to decision block 582 which, based on a configuration setting, determines whether the user is to confirm updates. If the user has required such confirmation, then decision block 582 routes execution, via YES path 583, to block 585. This latter block, when executed, notifies the user accordingly of the next scheduled update and requests the user to confirm it. Once the user so confirms the update, typically by clicking a displayed "OK" button with his mouse, execution proceeds to block 587 which logs an event, here a successful script-based update. Once the event is logged, execution exits from application 500. Alternatively, if the user has not requested update confirmation, then execution proceeds, via NO path 584 emanating from decision block 582, directly to block 587.

Detailed Description Text (49):
However, once the browser is able to open page INDEX.HTM at the update web site, the user then interacts, through the client PC and the browser, with the update web site to

download appropriate update files for product i and install the update files accordingly. As such, execution passes, via YES path 571 emanating from decision block 570, to block 573. Since actual downloading and installation of the update files will proceed through the web site and the browser and with no involvement of application 500 during that interval, block 573 merely waits until the user has closed the browser. Once this occurs, execution proceeds from block 573 to block 574. This latter block, when executed, automatically schedules the next update for product i. Thereafter, decision block 575 executes to determine, based on questioning the user and soliciting an appropriate response, whether the custom web site update completed. If the user indicates, typically through a mouse click on an appropriate button then appearing on the display, that the update completed, decision block 575 routes execution, via YES path 576, to decision block 577. This latter decision block determines whether the update script specifies that the user is to re-boot the client PC in order to complete the update. If such a re-boot is required, then this decision block routes execution, via YES path 579, to block 580. This latter block, when executed, displays a suitable notification to the user to re-boot the client PC. Once this notification occurs, execution proceeds to block 581. Alternatively, if such a re-boot is not required, then execution proceeds to block 581, via NO path 578 emanating from decision block 577. Block 581 updates the version number, stored in the O/S registry, for product i to that specified by the update script. Thereafter, execution proceeds to block 582 and so forth, as described above, after which execution ultimately exits from application 500. Here, the occurrence of a successful web-based update of product i will be logged as an event by block 587. If the custom web site update did not complete, then decision block 575 will route execution, via NO path 597 and path 568, directly to block 582 and so forth, as described above, after which execution ultimately exits from application 500. Here, the occurrence of an incomplete web-based update of product i will be logged as an event by block 587.

Detailed Description Text (55):
In particular, to utilize our inventive updating technique, an installation process for a product is modified, in the manner discussed below in conjunction with FIG. 11, to establish various entries under a single sub-key in the O/S registry. To minimize a burden placed on an installation program for this product, this program only creates entries under a single sub-key in the registry to effect product registration. However, use of our inventive technique with any item of software requires additional data beyond that which is created in the sub-key during product registration. This additional data is created and stored during product "detection". In that regard, once a user has installed a product and that product has registered itself for use with our inventive technique, then, if update application 15 is currently executing, "detection" can be initiated through use of configuration application 25 (see FIG. 3) and specifically by the user clicking on a "DETECT" button produced by this application and appearing on "common" configuration screen display 1410 shown in FIG. 14A. In this instance, the configuration application will determine the necessary data from existing data in the registry and store the former appropriately in both the registry and product configuration data 610. Once this occurs, then the product can be automatically updated through our technique, as well as can any of the update configuration data for that product. Rather than separately clicking on the "DETECT" button each time a new product has been installed, a user can install a number of products on his(her) client PC and then click the "DETECT" button once to collectively "detect" all of them and effectuate automatic updating therefor. In addition, product "detection" will also occur automatically whenever execution of update application 15 is initiated, such as manually through clicking on an icon or automatically through execution of a start-up group in a Windows O/S. Hence, the only time that the user needs to depress the "DETECT" button would be if, while application 15 is executing, the user decides to install a new product that registers itself with application 15 and the user does not want to close application 15 and re-launch its execution.

Detailed Description Text (56):
Specifically, once "detection" is initiated, then, as represented by line 631, product enumerator 630 first reads registry 247 and forms a list of all such products that have registered themselves for updating through the present invention, i.e., those products then having entries under particular product sub-keys in registry 247. Once this list is generated, product enumerator 630 passes, as symbolized by line 633, this list to configuration manager 635. The configuration manager, in turn, accesses the registry for all additional data which will be needed for automatically updating each of these registered products. The configuration manager then constructs, as symbolized by line 637, entries containing update configuration data for each such product, in a configuration sub-key in registry 247, so as to form a default working update configuration (see, e.g., for an illustrative "Cobra-DSL" product, those entries in

configuration sub-key 1250 shown in FIG. 12B which will be discussed below), in the registry, encompassing all such registered products. Lastly, for redundancy, configuration manager 635, as shown in FIG. 6, will also store, as symbolized by line 639, all the update configuration and product data existing in the registry in products configuration data 610.

Detailed Description Text (68):
FIG. 10 depicts a high-level flowchart of process 1000 for creating files on a network server, such as, e.g., server 70, which support our inventive software technique for updating a product.

Detailed Description Text (244):
Specifically, to begin de-registration, a user first launches, as indicated in block 1310, suitable "uninstall" software for a product which the user has elected to de-register. This software is conventional and is usually installed along with the product itself. The "uninstall" software will first eliminate all product files from local storage on the client PC as well as any registry entries, unrelated to product updating, which have been written to the O/S registry during product installation. Next, as indicated in block 1320, the "uninstall" software will delete its own product sub-key, which was created, as discussed above, during prior product installation in the HKEY.sub.-- LOCAL.sub.--
MACHINE.backslash.SOFTWARE.backslash.3Com.backslash.InstantUpdate.backslas h.Products key in O/S registry 247. Thereafter, as indicated in block 1330, the instant update client application 15 is either manually launched by the user or is automatically launched the next time the user logs onto the client, by e.g., execution of a start-up group (in the Windows 95 or NT O/S) containing an entry for application 15. Once this application is launched, configuration application 25 automatically removes, i.e., "cleans-up", all configuration entries and other settings in registry 247 under the key H.sub.-- KEY.sub.-- LOCAL.sub.--
MACHINE.backslash.SOFTWARE.backslash.3Com.backslash.InstantUpdate.backslas h.Configuration for all products that no longer have a corresponding sub-key in the registry products key. Alternatively, this same "clean-up" operation will occur whenever the user clicks on the "DETECT" button on configuration screen display 1410 (shown in FIG. 14A). Once this "clean-up" completely occurs through, e.g., block 1340, process 1300 is finished with the product being completely de-installed and de-registered. All related product information, including product and configuration data, are also automatically removed from products configuration data 610 (see FIGS. 6-9) through the operations in block 1340.

Detailed Description Text (247):
Screen display 1410 lists various installed products that have then registered themselves for use with updating application 500 and permits the user to select by, e.g., clicking on any one such product to set desired update configuration settings for the product through the remaining three screen displays. Illustratively, four such products are shown named as "Browning Lite", "Cobra DSL", "Instant Update" and Viper-DSL" with "Browning Lite" having been selected. Screen display 1410 also presents the user with the "DETECT" button, as described above, through which the user can initiate detection of all newly registered products. This screen display also permits the user to configure updating application 500 to log all events or not and to insert updating application 500 within a "Windows" start-up group such that this application will be automatically executed upon start-up of the Windows O/S at the client PC.

Detailed Description Text (248):
By selecting the "scheduling" tab which causes screen 1430 to be displayed, the user can schedule an automatic update for any product just selected through common screen display 1410 or initiate a update to start by depressing an "Update Now" button. Through "connection" screen display 1450, the user can indicate whether a connection to an update site is to be made through a LAN, specifically a TCP/IP connection, or through a dial-up connection, with for the latter the user specifying a phone book entry (containing a telephone number to dial for a remote network access port) and a name to be used for that connection. Lastly, through "general" screen display 1470, the user can indicate, for a selected product, whether (s)he is to confirm all updates (or not), specify a user name and password, where provided by the update site owner, for the update site for that product and, where appropriate, an update site address for that product (the user name, password and product site would override the defaults stored in the registry) so as to install, on a one-time basis, e.g., a "beta" release of that product from a different site than a default update site. Generally, a user name, password and product site address are not entered in screen display 1470 such that updating application 500 will utilize the defaults, stored in the registry, for

the selected product. However, the update site address would revert back to a registered (default) address whenever the user de-selects product site and user name boxes shown in screen display 1470.

Detailed Description Text (252):
Though we have described our invention for updating executable software, such as drivers, firmware, application programs, device configurations and O/S components, our invention can also be used to download and disseminate non-executable files, which are not necessarily update files, from an update site to client PCs. Such non-executable files can contain user information pertinent to an installed product, such as, e.g., new product information, help information, and/or user notifications from a product manufacturer. In that regard, a non-executable information file can be automatically downloaded with an instruction in the script to then ask the user whether (s)he wants to then view the file. If so, an instruction in the script can launch a local word processor or text editor on that file to properly display the information contained in the file to the user. Such a non-executable file can also be a stored profile, which provides configuration and/or other operational settings, for use with a particular product installed in networked client PCs. In this manner, a network administrator of, e.g., an enterprise-wide network, can store a common profile for a product--whether it is hardware or software (and whether it is updateable or not), and, through use of our present invention, propagate that profile and install it on each networked client PC that contains that product, hence automatically permitting the product to be uniformly configured on each such client. This, in turn, facilitates network-wide consistency and yields reduced network-wide support costs.

CLAIMS:

1. A method for updating software in a client-server environment through a network server, comprising the steps of, in a client computer:

(A) determining a country or a locale within which the client computer is situated and, in response thereto, constructing a file name for a file containing an update script for use in the country or the locale;

(B) issuing a request to the network server to download the file containing the update script to the client computer; and

(C) processing the script on the client computer so as to complete an update of the software, wherein the processing step comprises the steps of:

(C1) determining an operating system (O/S) then executing on the client computer; and

(C2) wherein the update script comprises at least one of O/S-specific and O/S-independent sections:

(C2a) if the O/S-independent section specifies a first update file name, extracting a first group of update file names from the O/S-independent section, the first group having at least one associated update file name; and

(C2b) if the O/S-specific section of the script corresponds to the O/S executing on the client computer and specifies a second update file name, extracting a second group of update file names from the O/S-independent section, the second group having at least one associated update file name; and

(C3) requesting a download to the client computer of update files from the network server corresponding to the first and second groups of update file names, to the extent at least one of the first and second groups of file names are specified in the O/S-independent and O/S-specific sections.

10. A method for updating software in a client-server environment through a network server, comprising the steps of:

in a client computer:

(A) determining a country or a locale within which the client computer is situated and, in response thereto, constructing a file name for a file containing an update script for use in the country or the locale;

(B) issuing a request to the network server to download the file containing the update script to the client computer; and

(C) processing the script on the client computer so as to complete an update of the software, wherein the processing step comprises the steps of:

(C1) determining an operating system (O/S) then executing on the client computer; and

(C2) wherein the update script comprises at least one of O/S-specific and O/S-independent sections:

(C2a) if the O/S-independent section specifies a first update file name, extracting a first group of update file names from the O/S-independent section, the first group having at least one associated update file name; and

(C2b) if the O/S-specific section of the script corresponds to the O/S executing on the client computer and specifies a second update file name, extracting a second group of update file names from the O/S-independent section, the second group having at least one associated update file name; and

(C3) requesting a download to the client computer of update files from the network server corresponding to the first and second groups of update file names, to the extent at least one of the first and second groups of file names are specified in the O/S-independent and O/S-specific sections; and

in the network server:

(D) downloading, in response to the request, the file containing the update script to the client computer.

18. Apparatus for updating software in a client-server environment through a network server, comprising:

a client computer having

a processor; and

a memory, connected to the processor, having computer executable instructions stored therein; and

wherein, in response to the stored instructions, the processor:

(A) determines a country or a locale within which the client computer is situated and, in response thereto, constructing a file name for a file containing an update script for use in the country or the locale;

(B) issues a request to the network server to download the file containing the update script to the client computer; and

(C) processes the script on the client computer so as to complete an update of the software through which the processor:

(C1) determines an operating system (O/S) then executing on the client computer; and

(C2) wherein the update script comprises at least one of O/S-specific and O/S-independent sections:

(C2a) if the O/S-independent section specifies a first update file name, extracts a first group of update file names from the O/S-independent section, the first group having at least one associated update file name; and

(C2b) if the O/S-specific section of the script corresponds to the O/S executing on the client computer and specifies a second update file name, extracts a second group of update file names from the O/S-independent section, the second group having at least one associated update file name; and

(C3) requests a download to the client computer of update files from the network server corresponding to the first and second groups of update file names, to the extent at least one of the first and second groups of file names are specified in the O/S-independent and O/S-specific sections.

26. Apparatus for updating software in a client-server environment through a network server, comprising:

a client computer having a processor; and

a memory, connected to the processor, having computer executable instructions stored therein; and

wherein, in response to the stored instructions, the processor:

(A) determines a country or a locale within which the client computer is situated and, in response thereto, constructing a file name for a file containing an update script for use in the country or the locale;

(B) issues a request to the network server to download the file containing the update script to the client computer; and

(C) processes the script on the client computer so as to complete an update of the software through which the processor:

(C1) determines an operating system (O/S) then executing on the client computer; and

(C2) wherein the update script comprises at least one of O/S-specific and O/S-independent sections:

(C2a) if the O/S-independent section specifies a first update file name, extracts a first group of update file names from the O/S-independent section, the first group having at least one associated update file name; and

(C2b) if the O/S-specific section of the script corresponds to the O/S executing on the client computer and specifies a second update file name, extracts a second group of update file names from the O/S-independent section, the second group having at least one associated update file name; and

(C3) requests a download to the client computer of update files from the network server corresponding to the first and second groups of update file names, to the extent at least one of the first and second groups of file names are specified in the O/S-independent and O/S-specific sections; and

the network server which:

(D) downloads, in response to the request, the file containing the update script to the client computer.

WEST

Generate Collection

L8: Entry 17 of 55 File: USPT Nov 21, 2000

DOCUMENT-IDENTIFIER: US 6151643 A
TITLE: Automatic updating of diverse software products on multiple client computer systems by downloading scanning application to client computer and generating software list on client computer

Application Filing Date (1):
19960607

Brief Summary Text (3):
The present invention relates to systems and methods for computer-based customer support, and more particularly, to systems, methods, and products for automatically updating software products from diverse software vendors on a plurality of end-user, client computer systems.

Brief Summary Text (12):
Accordingly, it is desirable to provide a system that automatically determines which software updates from numerous diverse software vendors are currently available, and which are applicable to a given user's computer system, and installs such user selected ones of such updates on the user's computer. Further, it is desirable to provide such a system without abridging the privacy of users by obtaining and storing system profile information.

Brief Summary Text (14):
In accordance with one aspect of the present invention, there is provided a system and method that automatically updates software components from numerous diverse software vendors on the computer systems of a plurality of end users. The system includes at least one database that stores software update information for a plurality of software products manufactured by diverse software vendors. The database is maintained by a service provider on a service provider computer system. Alternatively, the database may be maintained by a set of software vendors of the software products in association with the service provider. The software update information in the database specifies the software update program or files and their network location on the computer system of the software vendors, which computer systems are connected over the network to the service provider computer system. The database further stores information that describes an installation process for installing the software update on a user's computer.

Brief Summary Text (16):
On each user computer, or synonymously client computer, operating in accordance with one embodiment of the invention there is provided a client application that periodically connects over the network to the update database of the service provider computer system. The client application automatically downloads a portion of the database to the client computer, preferably to update a mirror of portions of database. From client database, the client application determines which software updates are applicable or relevant to the user's computer. This is preferably done by first determining the products that are installed in the client computer, and determining for each of these whether there is an update available for a more recent version of the software product than that installed on the client computer. The applicable software updates are identified to the user.

Brief Summary Text (20):
The above system allows numerous users to periodically and automatically update the software products on their computers from diverse software vendors through a single, update mechanism. The users need not invest the time and energy to identify currently available updates, nor engage in the potentially difficult process of manually (even electronically) obtaining and installing the software updates, and properly configuring their computer systems. Rather, all of the relevant information about the currently

available updates is maintained for subscribing users in the service provider's database. Further, the above system provides these benefits without directly storing the software updates themselves, which would be undesirable for the service provider due to vast memory requirements needed for handling software updates from hundreds, or potentially thousands of software vendors, and the difficulty of ensuring that all such software updates were current.

Detailed Description Text (9):
The update process 200 is typically initiated on the client computer 101. The user may manually initiate the process, or it may occur automatically, for example at preset periods, such as once a month. Alternatively, the process may be initiated by the service provider computer 102 prompting the client computer 101 at various intervals, or in response to particular events.

Detailed Description Text (10):
In each case, the user logs in 201 to the service provider computer 102 with the client application 104 in a conventional manner, providing a user ID, a password, and the like. This information may be manually entered by the user via the client application 104, or more preferably, stored within the client application 104, and automatically provided once a connection between the client computer 101 and service provider computer 102 is established. If the user is not registered, then the service provider computer 102 in conjunction with inputs by the user, registers 202 the new user of the system. FIG. 3 illustrates a basic user interface 300 for registering the user. The user identifies himself or herself by name 301 and selects a password 303. The user may also provide a mailing address 305 and a payment mechanism such as a credit card data 311, including a credit card number and expiration date, to pay for the services and for any for-fee software updates that the user may access in the course of using the service provided by the service provider computer 102. An email address 307 is entered to allow the service provider to contact the user by email. The user may select check box 309 to indicate that they want to be notified by email when new software updates are available for software products installed on their computer. When the registration process 202 is completed, the service provider computer 102 returns a unique registration number to the user. This number may be stored on the client computer 101 and used during subsequent logins to identify the user to the service provider computer 102.

CLAIMS:

37. A computer-implemented method of providing at least one software update on a client computer containing a plurality of software products, comprising:

maintaining on a service provider computer a database containing update information about a plurality of software products, the information including a network address for at least one software vendor server on the network that is separate from the service provider computer and contains at least one software update;

maintaining on the service provider computer a downloadable application, the application having a function for scanning the client computer to determine which software products reside on the client computer;

downloading the application to the client computer over a first communication path;

scanning the client computer with the application;

downloading a portion of the update information from the database to the client computer;

as a result of the scan, identifying software products that have been installed on the client computer for which updates are available;

ending client computer communications with the service provider computer;

selecting one identified software product to update;

downloading from the software vendor server to the client computer an update for at least one software product over a second communication path;

terminating communications between the software vendor server and the client computer; and installing the software update on the client computer;

wherein the service provider computer is not required to contain any software updates.

38. A method as recited in claim 37, wherein the application automatically downloads at least one software update on the client computer.

WEST

Generate Collection

L8: Entry 21 of 55 File: USPT Jul 25, 2000

DOCUMENT-IDENTIFIER: US 6094679 A
TITLE: Distribution of software in a computer network environment

Abstract Text (1):
A method of distributing software files resident on a network server to a network client. To effectuate the distribution, the network client issues an HTTP formatted request message to the network server which requests that certain software files resident on the network server be downloaded to the network client. The HTTP formatted request message may include information indicative of one or more of the operating system or processor architecture associated with the network client that the network server can use as an aide in determining which software files to return to the network client. The software files are bundled into a cabinet file by the network server and returned to the network client which, in turn, automatically unbundles the cabinet file, checks the authenticity of certain of the individual software files, and installs the software files in an appropriate memory location associated with the network client. In this manner, a world wide distributed printing solution is provided that is capable of working transparently on intranets and the Internet.

Application Filing Date (1):
19980116

Brief Summary Text (5):
A known method for distributing software is disclosed in commonly owned, pending U.S. patent application Ser. No. 08/641,087 U.S. Pat. No. 5,692,111, entitled "Automatic Installation Of Printers In A Distributed Environment", which is continuation of U.S. patent application Ser. No. 08/318,070 now abandoned filed Oct. 5, 1994 by Marby et al. and which is incorporated herein by reference in its entirety. In particular, the '087 application relates to the installation of a printer on a network client in a local area network which is performed in support of a point-and-print operation. The point-and-print capability allows a user to print on any printer available within the distributed network by merely selecting a printer and then requesting to print on that printer. The printer is installed by performing the steps of retrieving configuration software files from a network server that includes the target printer and automatically installing the retrieved software files on the network client. The retrieval and installation of the software files is performed transparently relative to the user of the network client.

Brief Summary Text (7):
In commonly owned, pending U.S. patent application Ser. No. 08/634,390 entitled "Method For Identifying And Obtaining Computer Software From A Network Computer", filed on Apr. 18, 1996 by Slivka et al., which is incorporated herein by reference in its entirety, a method for distributing software in a wide area network such as the Internet is disclosed. In particular, the '390 application discloses an update service which automatically inventories the network client to determine if any of its resident software files are out-of-date and/or in need of maintenance updates. If it is determined that software updates are required, the network server automatically downloads the needed software files in a compressed form to the network client along with a secure software installation application. The installation application downloaded by the network server is provided to cause the automatic installation of the downloaded software files on the network client.

Brief Summary Text (8):
While the system disclosed in the '390 application works for its intended purpose, it also suffers from various disadvantages. For example, no method is provided for automatically determining the form of the platform of the network client. Therefore, time may be wasted downloading software files that are incompatible with the platform of the network server. Additionally, since the installation application is provided by

the network server, there is also a risk that the installation application will not properly operate in connection with the particular platform of the network client, i.e., it will not execute, it will install the software files in the wrong memory locations, etc.

Brief Summary Text (13):
According to these needs, the present invention is generally directed to method of retrieving and downloading software files in a computer network having a network client linked to a network server in a wide area network, such as the Internet. The software is preferably hardware related software, such as printer driver/configuration files relating to a printer attached to the network server. The method allows for these software files to be retrieved by the network client, checked for authenticity, and automatically installed.

Detailed Description Text (20):
During the retrieval and packaging of the driver files, the server scripting component 76 is also used to create in step 128 a .BIN file which contains data pertaining to the server settings of the printer 50 and in step 130 a .DAT file which contains command lines which will be used by the Add Printer Wizard resident on the network client 20. As will be understood by those of skill in the art, the Add Printer Wizard is used to automatically install printers when provided the necessary files.

Detailed Description Text (30):
With reference to FIG. 8, once the cabinet file is returned to the network client 20, a file associating process of the operating system of the network client 20 will recognize the .IPP file extension provided to the cabinet file and, in response thereto, will cause the resident Printer Installer Application 82 (WPNINST.EXE) to start executing in step 136. The Printer Installer Application 82 functions to decompress and unpack the files in step 138 and, thereafter, to initiate in step 140 the verification of the executable software files which were supplied by the network server 49. For example, the verification process may be performed by calling Microsoft's Authenticode API which functions to present the user with a visual presentation of the digital signatures of the issuing agency of all the executable software files and, thereafter, inquire as to whether the user wishes to accept or reject the software files. If the user refuses to accept any one of the executable software files, the Printer Installer Application 82 preferably will delete in step 140 from the memory of the network client 20 all the software files returned by the network server 49 in response to the issued request message. If the downloaded software files are acceptable to the user, the Printer Installer Application 82 will cause in step 142 the Add Printer Wizard to execute which functions to install the printer on the network client 20. As more thoroughly described in pending U.S. patent application Ser. No. 08/641,087, entitled "Automatic Installation Of Printers In A Distributed Environment", which has been incorporated herein by reference in its entirety, the Add Printer Wizard utilizes the information contained within the .BIN and .DAT files, in connection with its routines, to initialize the printer settings to that of the print server while placing the software files returned by the network server 49 in the appropriate memory locations on the network client 20. Once completed, the printer 50 is effectively installed on the network client 20 and may be utilized to perform printing operations at the request of the network client 20.

CLAIMS:

1. In a network client in communication with a network server over a computer network, a method of distributing a software file resident on the network server to the network client, the method comprising the steps of:

receiving a command that the software file is to be downloaded to the network client;

automatically retrieving, in response to the command, information indicative of a platform of the network client;

appending to an HTTP formatted request message that requests that the software file be downloaded to the network client a data field comprising a representation of the information indicative of the platform of the network client;

issuing to the network server the HTTP formatted request message;

receiving from the network server in response to the issued HTTP formatted request message the software file; and installing the software file in an appropriate memory location associated with the network client.

7. In a network client in communication with a network server over a computer network, a method of distributing software files resident on the network server to the network client, the method comprising the steps of:

receiving a command that the software files are to be downloaded to the network client;

automatically retrieving, in response to the command, information indicative of a platform of the network client;

appending to an HTTP formatted request message that requests that the software files be downloaded to the network client a data field containing the information indicative of the platform of the network client;

issuing to the network server the HTTP formatted request message;

receiving from the network server in response to the issued HTTP formatted request message a cabinet file which contains the software files appropriate for the platform of the network client, the cabinet file having a file extension;

checking the file extension of the cabinet file and, as a function of the file extension, executing an appropriate installation application which is

component application of the operating system environment resident on the network client which performs the steps of extracting the software files from the cabinet file and installing the software files in an appropriate memory location associated with the network client.

11. A method of distributing software in a computer network comprising a network client linked to a network server having resident software files, the method comprising the steps of:

displaying a graphical user interface to a user on the network client which allows a user to issue a command that the software files are to be downloaded to the network client;

automatically retrieving, in response to the command, information indicative of a platform of the network client;

appending to an HTTP formatted request message that requests that the software files be downloaded to the network client a data field comprising data representative of the information indicative of the platform of the network client;

issuing from the network client to the network server the HTTP formatted request message;

collecting the software files at the network server as a function of the information wherein all executable software files have an associated digital signature;

packaging the collected software files into a cabinet file having a file extension;

sending the cabinet file to the network client from the network server; and

performing at the network client a check of the file extension of the sent cabinet file and, as a function of the file extension, executing an appropriate installation application which is component application of the operating system environment resident on the network client which performs the steps of extracting the software files from the cabinet file, initiating a check of the authenticity of each of the executable software files as a function of the digital signatures, and, if the executable software files are deemed to be acceptable, installing the software files in an appropriate memory location associated with the network client.

14. A computer-readable media having computer-executable instructions for use in performing a distribution of a software file resident on a network server to a network client, the instructions performing steps comprising:

automatically retrieving from the network client in response to a command to initiate the distribution of the software file information indicative of a platform of the network client;

appending to an HTTP formatted request message that requests that the software file be downloaded a data filed comprising data representative of the information indicative of the platform of the network client;

issuing to the network server the HTTP formatted request message; and

installing the software file in an appropriate memory location associated with the network client in response to a return receipt of the software file from the network server.

WEST

Generate Collection

L8: Entry 25 of 55 File: USPT Apr 4, 2000

DOCUMENT-IDENTIFIER: US 6047129 A
TITLE: Software updating and distribution

Abstract Text (1):
The present invention reduces complicated support and maintenance issues to a simple model. This model is called a procedure. A procedure includes two elements. The first element is called "criteria". Criteria are specified by the user and are used to create a filter that determines which workstations, workstation directories, workstation files, users, and/or file server directories are to be updated. The second element of a procedure is called an "action". Actions are used to specify what steps are to be taken automatically at the selected workstations or file server directories. Any type of process that can be carried out from the workstation keyboard can be automated using an action. One embodiment of the invention provides a system for updating software on at least one computer connected to a computer network, including hardware and software for defining at least one procedure, the at least one procedure including a set of criteria specifying a predetermined condition of the at least one computer hardware and software, responsive to the hardware and software for defining, for executing a predetermined action, such as a program, when the set of criteria are true.

Application Filing Date (1):
19980303

Brief Summary Text (3):
The present invention relates generally to the control of software and files on computer networks. More particularly, the present invention relates to a system for automatically distributing software to and updating software and files on workstations on a computer network.

Brief Summary Text (10):
Therefore, an object of the present invention is to provide a system for automatically distributing software from a file server to workstations located on a computer network.

Brief Summary Text (11):
Another object of the present invention is to provide a system for automatically updating, from a central location on a computer network, software installed in workstations connected to the computer network.

Brief Summary Text (14):
The present invention represents an entirely new class of network management software. It combines the classic concepts of software distribution with new approaches to file management. Automatic software distribution and installation, as well as automatic file updating and/or replacement are performed without requiring end-user intervention.

Brief Summary Text (16):
The second element of a procedure is called an "action". Actions are used to specify what steps are to be taken automatically at the selected workstations or file server directories. Some examples of actions might be to update certain files, delete certain files, edit specified files, or execute DOS batch commands.

Brief Summary Text (17):
The present inventor has realized that all software updating and distribution operations can be reduced to this two element paradigm. The user simply needs to specify, through criteria, the relevant characteristics of a workstation or file server directory that qualify the workstation or file server directory for updating. As soon as a workstation or file server directory meets the specified criteria, the specified action is automatically carried out. Any type of process that can be done from the

workstation keyboard can be automated using an action.

Drawing Description Text (6):
FIGS. 3 and 4 are flow charts illustrating the operation of the method of the present invention for automatically updating software;

Detailed Description Text (2):
For purposes of illustration only, and not to limit generality, the present invention will now be explained with reference to its use for updating software on PC workstations connected to a file server in a local area network. One skilled in the art will recognize that the present invention is not so limited and may be applied to updating software on any type of network having any type of computer connected to it.

Detailed Description Text (11):
Once the user has specified the criteria needed to determine which workstation or file server directory on the network will be updated, the user returns to the menu illustrated in FIG. 2A and chooses the Actions selection. Choosing Actions causes file server 12 to display the menu illustrated in FIG. 2K to the user. The Specify Files to Replace, Create, or Delete selection replaces workstation files with master copies from the network, creates new files by copying from master files, or deletes files from the workstation. The Text File Edit Commands selection allows modification of an existing file on the workstation by adding text to the beginning or end, searching for a specified string and adding new text after it, or searching for a specified string and replacing it with new text. The INI File Management selection allows the user to automatically perform most of the common modifications made on INI files including those INI files used by Microsoft Windows (such as Win.INI and System.INI). The DOS Batch Commands selection allows any DOS commands to be run on the workstation during the software updating process. The User Notification Message allows a message to be displayed to a user on display 22-1 of workstation 14-1, for example, informing the user that software on the workstation is about to be updated.

Detailed Description Text (21):
The present invention provides a number of advantages. It can automatically check workstation configurations on a computer network. If configurations of workstations change, this can be reported to the system administrator and/or automatically corrected in many cases. Additionally, when new workstations are added to the network, the system can automatically configure them to have a standard software configuration, thus avoiding the need to individually program each workstation as it is added to the network. For example, the present invention can be used to ensure that the CONFIG.SYS files of all workstations on the network are standardized.

Detailed Description Text (22):
The present invention advantageously allows computer network administrators to maintain the most up to date versions of their standard software applications, thus providing network users with the benefit of the newest features of programs. The invention also reduces the time during which a computer network is unavailable due to installing, updating, and fixing software that presently results in the inability of the network users to make use of the computer network. Furthermore, the present invention reduces the actual time that can be spent manually installing software across a network. Once a master copy of the software is loaded on file server 12, the present invention can automatically distribute and maintain the program on all the workstations connected to the computer network.

Other Reference Publication (5):
Ellison, C., "LAN Automatic Inventory", PC Magazine Jun. 30, 1992, v. 11, n. 12, p. 305(3).

Other Reference Publication (10):
Mather, Dan, Automated installation and updating of Windows-based Internet applications at James Madison University, 1995, pp. 207-209.

WEST

Generate Collection

L8: Entry 30 of 55 File: USPT Dec 28, 1999

DOCUMENT-IDENTIFIER: US 6009274 A
TITLE: Method and apparatus for automatically updating software components on end systems over a network

Abstract Text (1):
A method and apparatus for automatically updating software components in one or more agents (end systems) in a network. An ASU server generates a multicast request to agents within its network domain, identifying the newest, available versions of software components that may be installed on the agents. Agents compare installed versions with the newest versions and respond to the server request by indicating components that need to be updated. Components include network and non-network software as well as operating system (OS) software. The ASU server then transmits the requested components to the requesting agents in a self extracting compressed file. The file is installed and the components updated without rebooting system software.

Application Filing Date (1):
19970624

Brief Summary Text (2):
This invention relates to transmission of information between multiple digital devices on a network. More particularly, this invention relates to a method and apparatus for automatically updating and distributing executable files and components via a network in a distributed fashion.

Brief Summary Text (22):
Many prior art systems have been proposed to allow an IS staff person to manage and monitor network infrastructure remotely over a network. Such systems include Intel's LAN-desk, IBM's NetView, HP's OpenView, Norton Administrator, McAfee's ZAC or Novell's Network Management System (NMS). However, these systems generally rely on a full network protocol stack to be correctly running effectively on the remote ES in order to accomplish any remote file management operations. Often, however, ES system trouble or software updates results in one or more network protocol functions becoming non-operational. Under most prior art remote management systems, if any part of the remote ES network protocol stack is not working, the IS manager cannot access an ES through the network and must physically travel to the remote location to fix or diagnose the problem. Also, ISD management becomes extremely expensive in terms of ISD's keeping track of numerous different software modules on sometimes thousands of systems and of installing new versions or minor fixes of software on each system. It has therefore long been a desire of ISD managers and the owners of large numbers of ESs connected on a network to be able to automatically update software components over the network. Such updating ideally would be capable of updating both application software components, system software components, and even network components while the network is running. Such a system should also be able to automatically track version numbers of software in ESs and determine when update components must be downloaded.

Brief Summary Text (33):
What is needed in the art is a system for automatically installing and updating system level software components (OS), in addition to applications and agent software components, using an improved data link control protocol that requires fewer network resources than conventional protocols and which is less burdensome on network traffic. Also needed in the art is the ability to "pull" components down from a server (i.e., to update software remotely and automatically without requiring that agents automatically receive and load the newest versions of all software now available, but with the ability to support "push" technology if necessary).

Brief Summary Text (36):
The present invention is a method and apparatus for automatic software updating (ASU)

in a LAN. According to an embodiment of the invention, ASU agents, which are software or software plus hardware components, are placed within each (or a subset) of the ESs such as 50a-c, 51a-c, and 52a-g, connected to the LAN or within server machines. These agents may also be a part of a larger agent that includes other functions such as implementing prior art RMON functional groups an local packets as described in co-assigned patent applications, Ser. No. 60/040,876, Ser. No. 08/766,274, and Ser. No. 08/881/517.

Brief Summary Text (39):
According to one aspect of the invention, a method is provided for automatically updating software in a network including a server and an agent, comprising the steps of generating a server request, wherein said server request identifies the newest version level of a software component; generating an agent update request if the agent needs said newest version level of said software component; and updating the agent with said newest version level of said software component in response to said update request.

Brief Summary Text (41):
According to yet another aspect of the invention a method is provided for automatically updating software in a network including a server, a first agent and a second agent, comprising the steps of generating a multicast server request, wherein said multicast request identifies the newest version levels for a first software component and a second software component; generating a first agent update request if the first agent needs said newest version of one of said first and second software components; generating a second agent update request if the second agent needs said newest version of one of said first and second software components; and updating the first and second agents with said newest version levels of said first and second software components in response to said first and second update requests.

Detailed Description Text (8):
FIG. 4 is a block diagram of an embodiment of a ASU server according to the invention. Like the Agent, the ASU server loads automatically when the system starts and depends upon the same Desk Top Agent (DTA) services to exchange ASU protocol traffic with its Agents. The DTA is also used as a packet interface to allow the ASU server to monitor its own directed traffic as well as the broadcast and multicast traffic flowing within its sphere of management.

Detailed Description Text (19):
Improved ASU (Automatic Software Upqrader) Agent Updates

Detailed Description Text (20):
When ASU is widely deployed within a large institution, it is impractical to manually update agent software in each ES. Therefore, the system provides a mechanism for automatically updating the ASU agent and all of its components, including the DTA and NIC drivers, via the network. The mechanism is generalizable and may be used to automatically update any number of components, both network and non-network components, including system level (OS) software components. The mechanism for doing this is relatively simple within the ASU environment and takes advantage of the fact that all of the ASU's Agent components are dynamically loadable and unloadable; thus they can be replaced in operation without having to actually reboot the system.

Detailed Description Text (22):
Automatic Software Upgrade (ASU) is included in one embodiment of the invention because of the huge number of ASU agents that will be distributed throughout an enterprise's network. Without ASU, a user or administrator would need to manually install software every time a new upgrade for a component is released and this is prohibitively expensive for any MIS department. With ASU to automatically update any component at the ASU agent (end-system) from the ASU server, updating becomes a relatively painless process. According to a further aspect, this can also be used to help a user install other non-ASU-related components such as NIC drivers, system level (OS) software, etc. from one point per multicast segment or domain.

Detailed Description Text (28):
To help manage the process of Automatic Software Upgrade, another application, the Automatic Software Upgrade Manager (ASU Mgr.) is created. ASU Mgr. is a process running on a device with access to the network, and can be running on the same hardware as one of the ASU servers as described above. ASU Mgr. discovers all the ASU servers in the enterprise domain, through a poll/response or other mechanism. The ASU manager uses sockets to talk to each ASU server and can display a graphical listing on a management terminal of all the ASU agents in each ASU server's domain, for example. ASU Mgr.

allows a user to input and control the files to be updated from an ASU server to the ASU agents. The files are provided to the ASU Mgr. by a user, and the ASU Mgr. in turn uses FTP (or some other file transfer protocol) to transfer these files to ASU servers. Once the ASU server receives the files, the ASU server may transfer the files to the ASU agents. ASU Mgr. also allows a user to program or configure ASU server polling parameters (e.g., when to advertise to agents, when to accept and not accept responses, whether to advertise one component, several components or all components available, etc.) and update parameters (e.g., specifically including or excluding certain time intervals for updating). According to one embodiment of the invention, the ASU Mgr. provides a user with the ability to drill down into the graphical representation of each end node (ASU Agent) to find out information pertinent to the network, such as the current version of each ASU component, user name currently logged in, host name, etc.

Detailed Description Text (48):
The present invention performs updates in one embodiment by providing for two directory paths and updating the registry "on-the-fly" to point to the new update path when all updated components have been received from the network. In this embodiment of the current invention, the registry path is changed to point to a file not currently being used, but that will be used at the next reboot to cause an update to system components. This indirection provides a flexible way for the agent to perform updates on a continuous basis whereby the automatic updates can happen at any time. These updates are facilitated by the presence of the smart agent existing on the ES, always able to be polled by the ASU server to respond to update requests. Many prior art systems required users to log out periodically in order to run a network script to check for new system components. This can cause problems when the network incorrectly logs out a user that the network has incorrectly determined is not using network resources. Some prior art systems have proposed an agent residing on the ES to receive update packets, but these agents cannot perform version control as efficiently as the present invention.

CLAIMS:

1. A method of automatically updating software in a network including a server and an agent, comprising the steps of:

a) generating a server request, wherein said server request identifies the newest version level of a software component;

b) generating an agent update request if the agent needs said newest version level of said software component; and

c) updating the agent with said newest version level of said software component in response to said update request.

11. The network of claim 9, wherein said update request indicates a version level of said software component that is currently installed on said agent, and wherein said server compares said newest version level with said currently installed version level, and wherein said server updates said agent only if there is a discrepancy between said newest version level and said currently installed version level.

12. The network of claim 9, wherein said server updates said agent by transmitting the newest version of said software component to said agent, and wherein said newest version of said software component is installed by said agent without rebooting system software.

17. A method of automatically updating software in a network including a server, a first agent and a second agent, comprising the steps of:

a) generating a multicast server request, wherein said multicast request identifies the newest version levels for a first software component and a second software component;

b) generating a first agent update request if the first agent needs said newest version of one of said first and second software components;

c) generating a second agent update request if the second agent needs said newest version of one of said first and second software components; and

d) updating the first and second agents with said newest version levels of said first and second software components in response to said first and second update requests.

29. The network of claim 27, wherein said first and second update requests indicate version levels of said first and second software components currently installed on said first and second agents, and wherein said server compares said newest version levels with said currently installed version levels, and wherein said server updates said first and second agents only if there is a discrepancy between at least one of said newest version levels and said currently installed version levels.

30. The network of claim 27, wherein said server updates said first and second agents by transmitting the newest version of said software components to said first and second agents, and wherein said newest version of said software components are installed by said agents without rebooting system software.

WEST

Generate Collection

L8: Entry 31 of 55 File: USPT Dec 21, 1999

DOCUMENT-IDENTIFIER: US 6006034 A
TITLE: Systems and methods for automatic application version upgrading and maintenance

Application Filing Date (1):
19960905

Brief Summary Text (6):
Server-driven methods, however, are inherently unsuited for applications running on open-architecture networks, such as the Internet or intranet settings, in which the individual clients are difficult to access and control. The methods and systems of the present invention significantly improve the version updating process in a client-server environment in which such updating requires frequent and efficient deployment of the application components. In particular, the present invention shifts the version updating control to the individual client rather than the server, not only to reduce processing burden on the server but also to enable version updating in an open network environment, such as the Internet, where the source providing servers generally cannot control the remote clients. The methods of the present invention further adds security and protection from potential file corruption and undue delays during file transfer from a server to a client. By intelligently and automatically selecting to download and update only the needed and changed components of an application program, the present method alleviates the concerns of time and efficiency in any client-server network environment which requires highly dynamic application updates.

Brief Summary Text (12):
In the preferred embodiment, the frequency of the updating procedure is defined on a periodic basis or on an event-driven basis. For example, a predefined time interval, such as a day or week can be specified in the catalog, or by a user, and the application program is updated only on the first time the application is run in a specified time interval. In another embodiment, the application program on the client is automatically updated by an operating system or by a launcher program executed by a startup command on the client each time the client is booted up. In such an embodiment, the application program is caused to be updated regardless of whether the application program is executed at the client. Similarly, the client can be configured to update the application program periodically only as necessary to replace either outdated or corrupted components, or to delete any modules no longer needed, or to add any new modules.

Detailed Description Text (3):
In an open network architecture, such as the Internet and intranets, the centralized program updating is difficult due to the fact that the individual clients are not necessarily controlled by the server. In the Internet, for example, a Web server communicates with a remote client on an anonymous basis and cannot easily control the parameters of the client. FIG. 2A illustrates a preferred method of the present invention wherein a client 22 controls the process of a software upgrade in the client utilizing one or more servers 24 on a network. More particularly, in FIG. 2B, a software version upgrade can be initiated through executing an application program on the client 22. The execution command transmits a request signal 23 to the server 24 which holds the latest application components. In the preferred embodiment, the server responds by downloading a catalog of a list of the application components, each identified with the latest version number. Here, the server includes either a single computer or multiple computers or other servers networked together. The catalog file is processed by the client 22 to selectively identify and retrieve required components of the application program from the server 24.

Detailed Description Text (7):
Returning to FIG. 3A, the catalog file is retrieved at 302 from the server in response to a call from the client. In a preferred embodiment, as illustrated in FIG. 3C, the

client 22 of the present invention includes a launcher program 348 which is automatically activated when a user selects to run the application underlying icon 350 on a desktop window 346 of the client system 22. The launcher 348 serves as a proxy to the application program and communicates with the server 24 over a network to request a download of the catalog file.

Detailed Description Text (10):
The launcher program can be configured in different ways to accommodate users with different options to control and update the application program. In one embodiment, as shown in FIG. 4A, the launcher is a stand-alone program which can be executed through a desktop icon 400 on a client window. Selecting the icon executes the launcher program which provides a user with a dialog window to either select an existing application program to update at 401 or specify the network address of the catalog file for a new application at 402. Through the launcher dialog window(s), a user can select any application program either to execute, or to update and execute, or simply to update the components therein. In another embodiment, as shown in FIG. 4B, an icon directed to the application program can be installed on the client desktop window at 404. Selecting the icon automatically launches the launcher program in the background to begin an updating process. At 405, the launcher program in this embodiment is pre-configured with network address(es) of the catalog file as a parameter of the program.

Detailed Description Text (16):
Another aspect of the present invention relates to providing user flexibility at the client to control the updating procedures. In a preferred embodiment, as described in FIGS. 6A and 6B, a client can be configured to adapt a number of different updating schedules. In this embodiment, an application program may be selected to run either from a client desktop at 602 or during a boot sequence at 611, and the associated launcher is automatically executed to run a sequence of parameter checks. At 604, if the launcher is configured to run an update on a periodic basis, such as on a daily, weekly, or monthly basis, the launcher, through an internal calendar or clock, checks to determine if a new update is due with respect to the last update at 608, and, if so, executes an update process once within such a period at 609. If the update is specified during a boot sequence 611, the launcher is executed at 612 during such boot and performs the version update at 614 typically without executing the application program. In all the other configured situations the client defaults to run the launcher automatically in each application launch, unless otherwise specified by the user or in the catalog file. In the defaults, sending the launcher can be implemented as a component part of the client operating system to run invisibly in the background.

Detailed Description Text (17):
Yet another aspect of the invention relates to providing fully controlled client-server environment within an open network architecture, such as the Internet. In one preferred embodiment, as described in FIG. 7A, the client is a World Wide Web (Web) client, such as a Web browser, and the automatic version upgrading process is implemented to run fully within such a browser. In this embodiment, a hypertext link at 704 to an application program is provided within a hypertext markup language (HTML) document displayed on a Web page. Such a link is a uniform resource locator directed to a server site which makes available the catalog file for download through either the file transfer protocol (ftp) or the hypertext transfer protocol (http). The launcher program which receives and processes the catalog file is embedded (706) into the browser as a "plug-in" module or as a native browser control. A plug-in module is an integrated component of a browser which enables the execution of non-Web applications within the browser environment. A native browser control is built into the browser, and is hence more tightly integrated than any add-on modules on separate executables. At 708, the launcher is engaged to process the downloaded catalog file. Accordingly the components are downloaded to update the application program without leaving the Web browser at 710 and the program is executed at 712.

CLAIMS:

20. A method as claimed in claim 1 wherein the application program on the client is automatically updated on the client each time the client is booted up.

56. A system as claimed in claim 36 wherein the application program on the client is automatically updated by an operating system on the client each time the client is booted up.

67. A method of installing and automatically updating application programs on a plurality of client computers attached to a common network comprising the steps of:

storing various components of the application programs on one or more server computers attached to the same network with each server operating with standard protocol to automatically transmit a specified file in response to a standard file transfer request;

creating a catalog file which lists the names of all the required components of each of said application programs, and specifying for each component a current version identification and either a content of the component or a network address from which the component can be retrieved by the standard file transfer request;

storing the catalog file on one or more of said server computers; and

installing on each client computer a launcher program which operates as a proxy for each of said application programs and which executes steps for each application program comprising:

retrieving the current version of said catalog file and comparing the components and their version identifications to corresponding information in a second catalog of application components already stored on the client computer;

retrieving from their designated network addresses any components which said launcher program determines as either not present or having incorrect version identifications;

installing the retrieved components in a standard program component directory;

storing the retrieved catalog file to identified the components present on the client in a subsequent update; and

executing the application program.

68. A method as claimed in claim 67 further comprising:

creating on a World Wide Web site a link on a Web page to the catalog file; and

installing on each client computer the launcher program configured as a helper application in a Web browser to automatically execute whenever the link is selected to retrieve the catalog file.

82. A method of maintaining application program components on a network comprising:

maintaining on a server the application program, the program including components, each having a version identification, and maintaining a catalog of components with the version identifications;

maintaining the application program on a client;

in response to a call to the server from the client, causing the server to download the catalog to the client the catalog for the identified application program and, in the client, comparing the version identification between the components maintained on the server, indicated in the catalog, and the components maintained on the client;

updating the application program components on the client by downloading from the server to the client the selected components for which the version identifications do not match and replacing the selected components on the client; and

executing the updated application program on the client;

wherein the application program on the client is automatically updated on the client each time the client is booted up.

WEST

Generate Collection

L8: Entry 36 of 55 File: USPT Aug 17, 1999

DOCUMENT-IDENTIFIER: US 5940074 A
**** See image for Certificate of Correction ****
TITLE: Remote upgrade of software over a network

Application Filing Date (1):
19961122

Brief Summary Text (8):
Another problem encountered by computer users in general is that software applications tend to become outdated quickly. Accordingly, software suppliers periodically produce upgrades, which are often distributed in the same way that the original software was distributed, such as on magnetic or optical disks or other similar storage devices. However, the distribution of software upgrades on storage media such as these has disadvantages. For example, it is inconvenient and sometimes annoying for the user to have to repeatedly install software upgrades, which can be a time-consuming process. Further, a user may not be aware that an upgrade is available or necessary, or he may forget to obtain or install the upgrade. The failure to install an upgrade or a delay in installing an upgrade can be detrimental, since the upgrade may add valuable new features to the software or remedy a bug (error) in the software. Therefore, what is needed is a technique for allowing a software upgrade to be automatically provided over a network in a manner which requires little or no effort on the part of the user.

Detailed Description Text (15):
The box 10 includes application software which, when executed by a processor in the box 10, provides the user with a graphical user interface by which the user can access the WebTV network services and browse the Web. The application software is automatically executed upon application of power to the box 10.

Detailed Description Text (25):
As indicated above, the browser software can be upgraded or reconfigured by downloading to the box replacement software or data transmitted from a server via the Internet or via direct phone connection. The programmable nature of flash memory 22b and its ability to retain programmed information in the absence of power are used to advantage in performing such an upgrade or reconfiguration. As will be explained in greater detail below, replacement software or data transmitted from the WebTV server 5 (or another server designated by the WebTV server 5) is automatically written into the flash memory 22b in the WebTV box 10. In addition, the flash memory can be used to store various resources downloaded form the Web, such as Java applets (programs), so that such resources will be retained in the event of loss of power to the client system 1. Note that the present invention does not necessarily require use of a flash memory for these purposes; other forms of programmable non-volatile memory may be used, such as an electrically-erasable programmable ROM (EEPROM).

Detailed Description Text (30):
The reset routine of FIG. 6 is performed any time the client system 1 is reset during operation or any time power to the box 10 is turned on. In one embodiment, the routine of FIG. 6 is performed by execution of start-up instructions stored in the mask ROM 22a. In step 601, the validity of all contents of the flash memory 22b (i.e., program instructions and data) are verified using a conventional checksum technique. If the contents are valid (step 602), then the normal start routine is performed in step 603. If the contents of flash memory 22b are not valid (i.e., are corrupted or otherwise found to represent an inconsistent state), then an error download routine is performed in step 604. In the error download routine, some or all of the corrupt information in the flash memory is replaced by correct information downloaded from the Internet 3. The error download routine is described further below. Thus, the WebTV system allows errors in the programming or data to be detected and automatically corrected by performing the error download routine, without intervention by the user of the WebTV client system 1.

Detailed Description Text (31):
FIG. 7 illustrates a routine by which an upgrade of the Web browser is initiated. During a normal start-up (i.e., when no error was detected upon reset), the client system 1 automatically connects to the WebTV server 5 in step 701. Generally, this connection is made via the modem pool 2 by executing a connection script. If an upgrade is determined to be available in step 702, and the upgrade is designated as mandatory in step 703, then the server sends a command to the client system 1 to cause a download request to be written into the flash memory 22b of the client system 1 in step 704. The client system 1 is then commanded by the WebTV server 5 in step 705 to reset according to the routine of FIG. 6. If an upgrade is determined to be available in step 702, and the upgrade is not designated as mandatory in step 703, then the client system 1 prompts the user in step 706 to either accept or decline the upgrade. If the upgrade is accepted in step 707, then the client system 1 resets in step 705 according to the routine of FIG. 6. If not, the routine ends.

Detailed Description Text (34):
Referring still to FIG. 8, if the local connection script is found in flash memory 22b (step 802), then in step 803 the client system uses that connection script too connect to the WebTV server 5 via the local modem pool 2. Assuming such connection is made, a determination is then made in step 804 as to whether an IP (Internet Protocol) address, a port, and path information for the upgrade is stored in the flash memory 22b. The WebTV server 5 may provide such information to the client system 1 if the upgrade is to be downloaded from a server other than a default server, which may be the WebTV server 5. If such information has been provided, then the client system 1 connects to the specified server in step 805, requests the specified file in step 806, and initiates downloading of the file in step 807. If an IP address, port, and path are not found in flash memory 22b, then in step 810 the client system 1 connects to the default server using a default IP address, port, and path stored in mask ROM 22a. A default upgrade file is then requested using this stored information in step 811. In step 811, the client system 1 also indicates to the default server which version of software it is currently running, so that the default server can determine the proper default file for that client system. Downloading of the default file is initiated in step 807 after the client system has requested a file. As mentioned above, downloading to the client system 1 occurs via the Internet 3 via the modem pool 2. Once downloaded, the information is automatically decompressed (if compression was applied) and written into flash memory 22b by the client system 1.

Detailed Description Text (35):
Hence, an optional upgrade is performed automatically without any input from the user of the client system 1, other than the user's answering a prompt on whether to accept the upgrade. A mandatory upgrade is performed without any input from the user and, in fact, can be performed without informing the user, if desired.

Detailed Description Text (36):
FIG. 9 illustrates a routine by which an error in the client system's programming or data can be automatically corrected, such as when an error or other inconsistent state is found during the reset routine (FIG. 6). It will be seen that such correction occurs automatically without any input from the user and, in fact, can occur without informing the user, if desired. Initially, in step 901 the client system 1 connects to the server 5 directly using the default toll-free number stored in mask ROM 22a. Once connected, the client system 1 obtains a local connection script from the server 5. The client system 1 then disconnects from the server 5 and then reconnects to the server 5 via the local modem pool 2 using the local connection script. The client system 1 then further establishes a connection to the default server in step 903 using the default IP address, port, and path stored in mask ROM 22a. In step 904, the client system 1 requests the default upgrade file from the default server using the default IP address, port, and path. In step 904, the client system 1 also indicates to the default server which version of software it is currently running, so that the default server can determine the proper default file for that client system. In step 905, the requested file is downloaded over the Internet 3. The downloaded file is then written into flash memory 22b.

CLAIMS:

1. In a networked computer system which includes a plurality of server systems each of which is connected to a network infrastructure which provides access through said servers to a plurality of sites, and the networked computer system including at least one client system which is connected either directly or logically to one or more of

said servers, and each client system comprising a conventional television monitor on which to display information retrieved from of an accessed site, and an electronic unit including program instructions stored in any combination of a mask ROM, RAM, flash memory, mass storage device or CPU memory, and wherein the electronic unit includes a CPU for executing said program instructions so as to control said television monitor in order to provide a user of the client system with a graphical user interface by which the user can access the network infrastructure to browse said sites or otherwise access said network infrastructure, a method for automatically downloading to said CPU of the electronic unit software or data in order to replace or upgrade said program instructions used by the CPU to access the network infrastructure, without the need for user intervention, the method comprising the steps of:

initializing the client system by powering on the electronic unit and automatically connecting the client system to at least one of said servers;

automatically and without the need for user intervention, checking the validity of the stored program instructions to ascertain the existence of a corrupted state in the stored program instructions of the electronic unit of the client system, and if the validity of the stored program instructions is not verified, then automatically and without the need for user intervention, replacing the stored program instructions by downloading to said electronic unit a replacement for the program instructions from said at least one of said servers;

automatically and without the need for user intervention, checking said at least one sever for the existence of an upgraded version of the program instructions, and if the program instructions of the electronic unit are out of date, then automatically and without the need for user intervention, either (1) replacing the stored program instructions of the electronic unit with the upgraded program instructions downloaded from said at least one of said servers if the upgrade is designated as mandatory at said at least one server, or (2) if the upgrade is not mandatory, prompting the user to decide whether to accept the upgrade from said at least one server; and

using the upgraded or replaced program instructions to control said television monitor in order to provide a user of the client system with a graphical user interface by which the user can access the network infrastructure to browse said sites or otherwise access said network infrastructure through one or more of said servers.

7. In a networked computer system which includes a plurality of server systems each of which is connected to a network infrastructure which provides access through said servers to a plurality of sites, and the networked computer system including at least one client system which is connected either directly or logically to one or more of said servers, and each client system comprising a conventional television monitor on which to display information retrieved from of an accessed site, and an electronic unit including program instructions stored in any combination of a mask ROM, RAM, flash memory, mass storage device or CPU memory, and wherein the electronic unit includes a CPU for executing said program instructions so as to control said television monitor in order to provide a user of the client system with a graphical user interface by which the user can access the network infrastructure to browse said sites or otherwise access said network infrastructure, a computer program product for use by said electronic unit to implement a method for automatically downloading to said CPU of the electronic unit software or data in order to replace or upgrade said program instructions used by the CPU to access the network infrastructure, without the need for user intervention, the computer program product comprising:

a computer readable medium for carrying computer program code means for implementing said method; and

said computer program code means comprising:

code means for initializing the client system when powering on the electronic unit and automatically connecting the client system to at least one of said servers;

code means for automatically and without the need for user intervention, checking the validity of the stored program instructions to ascertain the existence of a corrupted state in the stored program instructions of the electronic unit of the client system, and if the validity of the stored program instructions is not verified, then automatically and without the need for user intervention, replacing the stored program instructions by downloading to said electronic unit a replacement for the program instructions from said at least one of said servers;

code means for automatically and without the need for user intervention, checking said at least one server for the existence of an upgraded version of the program instructions, and if the program instructions of the electronic unit are out of date, then automatically and without the need for user intervention, either (1) replacing the stored program instructions of the electronic unit with the upgraded program instructions downloaded from said at least one of said servers if the upgrade is designated as mandatory at said at least one server, or (2) if the upgrade is not mandatory, prompting the user to decide whether to accept the upgrade from said at least one server; and

replacing the stored program instructions of the electronic unit with the upgraded program instructions downloaded from said at least one of said servers; and

code means for using the upgraded or replaced program instructions to control said television monitor in order to provide a user of the client system with a graphical user interface by which the user can access the network infrastructure to browse said sites or otherwise access said network infrastructure through one or more of said servers.

13. In a networked computer system which includes a plurality of server systems each of which is connected to a network infrastructure which provides access through said servers to a plurality of sites, and the networked computer system including at least one client system which is connected either directly or logically to one or more of said servers, and each client system comprising a conventional television monitor on which to display information retrieved from of an accessed site, and an electronic unit including program instructions stored in any combination of a mask ROM, RAM, flash memory, mass storage device or CPU memory, and wherein the electronic unit includes a CPU for executing said program instructions so as to control said television monitor in order to provide a user of the client system with a graphical user interface by which the user can access the network infrastructure to browse said sites or otherwise access said network infrastructure, a computer program product for use by at least one of said servers to implement a method for automatically downloading to said CPU of the electronic unit software or data in order to replace or upgrade said program instructions used by the CPU to access the network infrastructure, without the need for user intervention, the computer program product comprising:

a computer readable medium for carrying computer program code means for implementing said method; and

said computer program code means comprising:

code means for determining at said at least one server when a client system has been initialized and automatically connected to at the least one of said servers;

code means for automatically and without the need for user intervention, responding to a request from the client system after the client system has automatically checked the validity of the stored program instructions to ascertain the existence of a corrupted state in the stored program instructions of the electronic unit of the client system, and if the validity of the stored program instructions is not verified, then automatically and without the need for user intervention, downloading to said electronic unit a replacement for the program instructions from said at least one of said servers; and

code means for automatically and without the need for user intervention, advising said client system of the existence of any upgraded version of the program instructions contained at said at least one server, and if the program instructions of the electronic unit are out of date, then automatically and without the need for user intervention, either (1) replacing the stored program instructions of the electronic unit with the upgraded program instructions downloaded from said at least one of said servers if the upgrade is designated as mandatory at said at least one server, or (2) if the upgrade is not mandatory, prompting the user to decide whether to accept the upgrade from said at least one server.

WEST

Generate Collection

L8: Entry 41 of 55 File: USPT Nov 10, 1998

DOCUMENT-IDENTIFIER: US 5835911 A
**** See image for Certificate of Correction ****
TITLE: Software distribution and maintenance system and method

Abstract Text (1):
A number of sets of software may be systematically distributed and maintained via a network connecting many vendors and users of client/server software. A client program in a user computer detects when software subject to maintenance is activated and transmits an inquiry over the network to the software vendor's computer for information on the current version of the software. The server program compares data in the inquiry with data relating to the latest version of the software and returns update instruction information and updated software if appropriate. The client program automatically updates the software to the latest version according to the update instruction information when it is received. The client program can also send inquires at predetermined times, or in response to a user command. The inquiry can include a request for purchase information in which case the server checks qualifications of the user, processes the inquiry according to vendor management data and returns the requested software, if appropriate. Other inquiries can also be made in response to user commands or automatically, e.g., to obtain information on the most recent version and transmission of data from client to server in response to an abnormal termination of client software.

Application Filing Date (1):
19950821

Brief Summary Text (21):
(e) Automatic Download

Brief Summary Text (22):
This method is used for an electronic notice board. In this method, a large amount of news information is stored in a central computer, etc., and a terminal computer can access the central computer through the network periodically at intervals and automatically read new information in a specified category if newly stored (download).

Brief Summary Text (24):
The software which is automatically downloaded is stored as ordinary data, and does not directly update or manage the software currently used in the user computer. That is, the above described automatic download system does not manage the software currently used by a user.

Brief Summary Text (28):
To attain this, the secondary storage centers periodically access the primary storage center, and detect and read newly stored/updated information only. This method is different from the above described automatic downloading method (e) in obtaining information in plural categories by switching for each category the functions of the primary storage center and the secondary storage centers among the central computers in a network.

Brief Summary Text (31):
If a fault occurs in a user computer, a software (or hardware) vendor directly operates user software through a network at a user's request (or through automatic communications over the network) to detect the reason for the fault and recover from the fault.

Brief Summary Text (33):
That is, the process is different from an automatic supply of maintenance and update services by automatic software for a number of users.

Brief Summary Text (35):
The client/server method, in which information is communicated between the client and server software and a client operates server software, is well known and has been used to realize the above listed technologies (b)-(f). No systems, however, have been developed so far with this method to automatically update software already distributed to a user, by simply updating the software in the vendor computer.

Brief Summary Text (53):
A comparatively simple system to solve these problems can be a system of automatically updating an object software in a computer of each user to the latest version by use of a client/server method A single object program is managed by client programs in a number of user computers and is supported by a server program which manages the software library of the object software in a vendor computer.

Brief Summary Text (78):
A feature of the present invention resides in a software distribution/maintenance system having a plurality of user computers and a vendor computer connected to the plurality of user computers through a network to manage and automatically update over the network a set of object software sent and stored in the user computers from the vendor computer through the network, comprising: first process means in the user computers; second process means in the vendor computer; and object software library in the vendor computer, wherein said first process means sends current configuration information of the object software stored in the user computers to said second process means of the vendor computer through the network to inquire the latest configuration, receives an answer from said second process means, and updates the object software stored in the user computers according to an instruction in a received answer; and

Brief Summary Text (80):
Another feature of the present invention resides in a software distribution/maintenance system having a plurality of user computers and a vendor computer to manage and automatically update object software provided for the plurality of user computers from the vendor computer thorough a network, comprising: first process means in each of the user computers; and

Detailed Description Text (7):
The above described configuration is used in an automatic remote software update system in which a single object software is used as a principle. Furthermore, according to the second embodiment, each user computer is usually provided with plural sets of object software, and each vendor computer is usually provided with a plurality of software libraries.

Detailed Description Text (11):
According to the present invention, the server program in the vendor computer returns the answer message in response to the software distribution/maintenance request message from the client program, thereby automatically distributing/maintaining the object software as described above.

Detailed Description Text (17):
To solve the above described problems, a software distribution/maintenance system according to the present invention comprises, as shown in FIG. 1, a plurality of the user computers 1-1, . . . , 1-n and the vendor computer 3 connected to the plurality of user computers 1-1, . . . , 1-n through the network 2, and manages and automatically updates over a network the object software 1a stored in the user computers 1-1, . . . 1-n from the vendor computer 3 through the network 2.

Detailed Description Text (20):
Furthermore, according to the present invention, if the operation of the object software 1a abnormally terminates in the user computers 1-1, . . . , 1-n, the first process units 1b in the user computers 1-1, . . . 1-n automatically inform over the network 2 the vendor computer 3 of an abnormal termination and its state.

Detailed Description Text (21):
The first process unit 1b according to the present invention automatically informs the vendor computer 3 of the abnormal termination, the instruction causing the abnormal termination and the reason for the abnormal termination, a series of instructions which invoked the instruction causing the abnormal termination, and the environment of the software/hardware used when the abnormal termination is detected.